**VI Semester B.Tech. (IT)**

**ICT 3266:Internet Tools & Technology Lab**

**PROJECT REPORT**

# INVENTORY MANAGEMENT SYSTEM

***submitted by***

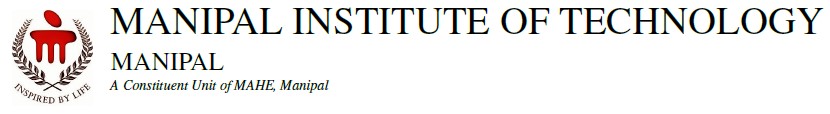
Aadil Faisal (210911056, Roll No: 14, IT-A)

*Under the guidance of*

Ms. Chetna Pujari (Lab Faculty 1) Dr. Raghavendra S (Lab faculty 2)

Assistant Professor-Senior Scale Associate Professor-Senior Scale Department of I&CT Department of I&CT

Manipal Institute of Technology Manipal Institute of Technology Manipal, Karnataka, India Manipal, Karnataka, India



April 2024

|  |
| --- |
| **Contents** |
| **1.Introduction 3**  **2.Project Objective 3**  **3.Design 4**  **4.Implementation Details 6**  **5.Testing And Validation 12**  **6.Performance Evaluation 13**  **7.Conclusion 14**  **8.Future Scope 14** |

## 1. Introduction

In the fast-paced realm of retail, the Inventory Management System emerges as a pivotal innovation, addressing the dynamic needs of businesses for streamlined and efficient inventory control. Our system provides a seamless platform tailored to simplify the intricate process of managing stock, optimizing procurement, and ensuring product availability. With a core emphasis on enhancing operational efficiency through intuitive interfaces, this project aims to modernize conventional inventory management practices, making them more accessible and user-centric. Going beyond mere stock tracking, the system boasts comprehensive features including detailed product catalogs, real-time inventory monitoring, and robust authentication mechanisms, guaranteeing data integrity and scalability to adapt to future trends in the retail landscape.

## 2. Project Objectives

**User Module Objectives**

* **Seamless Navigation:** Develop an intuitive user interface that enables customers to effortlessly search for Items, browse categories, and add items to their cart, enhancing the overall shopping experience**.**
* **Convenient Ordering:** Integrate a user-friendly ordering system that allows customers to easily select quantities, specify preferences (e.g., size, brand), and schedule delivery or pickup, ensuring a smooth and efficient shopping process.
* **Seamless Checkout Process:** Streamline the checkout process by offering multiple payment options, ensuring secure transactions, and providing order tracking functionalities to keep users informed about the status of their purchases.

**Admin Module Objectives**

* **Efficient Product Management**: Provide tools for admins to easily add, update, or remove items, categories, and inventory information, facilitating accurate and up-to-date content management.
* **Ordering and Transaction Oversight:** Implement a dashboard for monitoring and managing all ordering transactions, including order fulfillment status, payment processing, and handling returns or refunds, to streamline operations.
* **Inventory Management:** Equip the admin module with features for managing inventory levels, tracking stock movements, and generating reports to optimize stock replenishment and prevent stockouts.
* **Security and User Management:** Implement robust tools for managing admin accounts and roles, enforcing access controls to restrict unauthorized actions, and ensuring compliance with data protection regulations to safeguard sensitive information.

**3.Design Architecture:**

A screenshot of a computer

Description automatically generated

The architecture of the project revolves around a Flask-based web application for managing inventory, user accounts, and orders. Here's a brief explanation of the architecture based on the provided tables:

1. Flask Application: The core of the application is built using the Flask framework, which follows the Model-View-Controller (MVC) pattern. Flask provides routing, request handling, and templating capabilities, making it suitable for building web applications.
2. Database Layer:
   * SQLite Database: The application uses an SQLite database to store persistent data. SQLite is a lightweight, serverless database engine that is well-suited for small to medium-sized web applications.
   * SQLAlchemy ORM: SQLAlchemy is an Object-Relational Mapping (ORM) library for Python, used to interact with the database in an object-oriented manner. It provides a high-level interface for defining database models and executing database queries.
3. Model Layer:
   * User Model: Represents user accounts in the system. It stores information such as phone number, address, password hash, and first name. The UserMixin from Flask-Login is used to extend the User class with user authentication functionalities.
   * Product Model: Represents products available in the inventory. It stores details such as product ID, name, price, quantity available, and image URL.
   * Cart Model: Represents the user's shopping cart, linking users and products in a many-to-many relationship. It stores the user's phone number (cart\_id), the product ID (product\_id), and the quantity of each product in the cart.
   * Orders Model: Represents orders placed by users. It stores details such as the order ID, customer's phone number (order\_customer), date and time of the order (order\_date\_time), order summary, order amount, and delivery address.
4. Controller Layer:
   * The controller layer consists of the Flask routes and views that handle incoming requests, interact with the database using SQLAlchemy, and render HTML templates to generate dynamic web pages.
   * Routes are defined to handle user authentication, product catalog management, cart management, order processing, and other application functionalities.
5. View Layer:
   * HTML templates are used to define the presentation layer of the web application. Flask's templating engine allows embedding Python code within HTML templates to dynamically generate content.
   * Templates are rendered by Flask views and returned as HTTP responses to the client's browser. The client interacts with the web pages to browse products, add items to the cart, place orders, and perform other actions.

Overall, the architecture follows a typical client-server model, where the Flask application serves as the backend server, handling requests from clients (web browsers), interacting with the database, and generating dynamic HTML content to be displayed to users. The application's modular structure facilitates scalability, maintainability, and extensibility as the project evolves.

**4. Implementation Details:**

The implementation of the inventory management system involved several key components and technologies. Below are the details of each component:

* **Frontend Development**: The frontend of the system was developed using HTML, CSS, and JavaScript. HTML was used for structuring the web pages, CSS for styling the elements, and JavaScript for client-side interactivity such as form validation and dynamic content loading.
* **Backend Development**: Python Flask framework was used for developing the backend of the system. Flask provided a lightweight and flexible framework for building web applications. It facilitated routing, request handling, and integration with the frontend.
* **Database Management**: SQLAlchemy, a Python SQL toolkit and Object-Relational Mapper (ORM), was utilized for managing the database. SQLAlchemy abstracted away the complexities of SQL queries and provided a Pythonic way to interact with the database. SQLite was chosen as the database engine for its simplicity and ease of use.
* **User Authentication**: User authentication was implemented using Flask-Login, an extension for Flask that manages user sessions. It provided features like login, logout, and session management, ensuring secure access to the system.
* **Admin Page**: An admin page was created to allow administrators to add products to the catalog. This page was accessible only to authenticated admin users and provided a form for adding product details such as name, quantity, price, and image.

A screenshot of a computer

Description automatically generated

Figure 1: Admin Page

A black and grey striped background

Description automatically generated

Figure 2: Login Page

A black and white striped background

Description automatically generated

Figure 3:Signup Page

A screenshot of a computer

Description automatically generated

Figure 4: Homepage

A screenshot of a video game

Description automatically generated

Figure 5:Cart Page

A screenshot of a computer

Description automatically generated

Figure 6: Profile Dashboard

**5) Testing and Validations:**

Unit Testing:

* **Test Cases**: We developed comprehensive unit tests to validate the functionality of each module, including user authentication, product management, cart operations, and order processing.
* **Test Coverage**: Our unit tests cover critical code paths and edge cases to ensure robustness and reliability.

Integration Testing:

* **End-to-End Testing**: We conducted integration tests to validate the interactions between different components of the system, including database operations, Flask routes, and HTML rendering.
* **User Scenarios**: We simulated various user scenarios to verify that the application behaves as expected under different conditions.

**6. Performance Evaluation:**

Scalability:

* **Database Performance**: We assessed the performance of the SQLite database under various load conditions to ensure scalability as the application's user base grows.
* **Flask Performance**: We monitored the performance of Flask routes and views to identify potential bottlenecks and optimize request processing times.

Resource Utilization:

* **Memory Usage**: We evaluated the memory footprint of the application to optimize resource utilization and ensure efficient memory management.
* **CPU Utilization**: We monitored CPU utilization to identify CPU-bound tasks and optimize code execution for better performance.

**7. Conclusion:**

Achievements:

* **Successful Implementation**: We successfully developed and deployed an inventory management system using Flask, SQLAlchemy, and SQLite.
* **Feature Completeness**: The application meets the specified requirements, including user authentication, product management, cart functionality, and order processing.

Lessons Learned:

* **Enhanced Skills**: Through this project, we gained practical experience in web development using Flask and database management using SQLAlchemy.
* **Best Practices**: We learned and applied best practices in software design, development, and testing, contributing to our professional growth.

**8. Future Scope:**

Feature Enhancements:

* **Additional Functionality**: We plan to enhance the application with additional features such as user reviews, inventory alerts, and discount promotions.
* **User Interface Improvements**: We aim to improve the user interface by incorporating modern design principles and responsive layouts for better user experience.
* **Performance Optimization**: We will continue to optimize the application's performance by fine-tuning database queries, caching frequently accessed data, and implementing asynchronous processing where applicable.

Scalability:

* **Horizontal Scaling**: As the user base grows, we will explore options for horizontal scaling by deploying the application across multiple servers and load balancing incoming requests.
* **Database Sharding**: We may consider database sharding techniques to distribute data across multiple database instances and improve scalability and performance.
* Integration:
* **Third-Party Integrations**: We plan to integrate the application with third-party services such as payment gateways, shipping providers, and analytics platforms to enhance functionality and provide a seamless user experience.
* Community Engagement:
* **Open Source Contribution**: We intend to contribute back to the open-source community by sharing our code, documenting best practices, and participating in relevant forums and discussions.
* **User Feedback**: We will actively solicit feedback from users to identify areas for improvement and prioritize feature requests based on user needs and preferences.
* By incorporating these updates into your project report, you can provide a comprehensive overview of your inventory management system, its implementation details, testing and validation processes, performance evaluation, conclusions, and future scope.
* Top of Form